NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

FACILITY LOCATION USING CROSS DECOMPOSITION

by

Leroy A. Jackson

December 1995

Thesis Advisor: Robert F. Dell

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 14, 1995	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE FACILITY LOCATION USING CROSS DECOMPOSITION			5. FUNDING NUMBERS
6. AUTHOR(S) Jackson, Leroy A.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Determining the best base stationing for military units can be modeled as a capacitated facility location problem with sole sourcing and multiple resource categories. Computational experience suggests that cross decomposition, a unification of Benders Decomposition and Lagrangean relaxation, is superior to other contemporary methods for solving capacitated facility location problems. Recent research extends cross decomposition to pure integer programming problems with explicit application to capacitated facility location problems with sole sourcing; however, this research offers no computational experience. This thesis implements two cross decomposition algorithms for the capacitated facility location problem with sole sourcing and compares these decomposition algorithms with branch and bound methods. For some problems tested, cross decomposition obtains better solutions in less time; however, cross decomposition does not always perform better than branch and bound due to the time required to obtain the cross decomposition bound that is theoretically superior to other decomposition bounds.

14. SUBJECT TERMS cross decomposition, facility location, unit stationing			15. NUMBER OF PAGES 64
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500 | Standard Form 298 (Rev. 2-89)
Prescribed by ANSI S.d.. 239-18 298-102

EXECUTIVE SUMMARY

The problem of determining the best stationing plan for military units can be modeled as a capacitated facility location problem with sole sourcing and multiple resource categories. The capacitated facility location problem accomplishes two goals. It determines which facilities to open and how to allocate customer demand among the open facilities. For unit stationing this equates to which bases to retain and which units to assign to bases. Sole sourcing requires that all customer demand be allocated to one facility. For the base stationing application this requirement means that a unit is stationed at a single base. In capacitated facility location problems with multiple resource categories the customers require more than one type of resource from the facilities. For unit stationing problems the resources include those that are consumed by the presence of the unit, such as barracks and motor pools, and those that are shared by units, such as ranges.

The most common techniques for solving models like the capacitated facility location problem with sole sourcing are branch and bound methods. Some problem instances of the capacitated facility location model with sole sourcing require many hours to solve using branch and bound. For some instances it is not practical to solve the problem optimally, but it is possible to find a feasible solution and a lower bound on the cost of the optimal solution. Decomposition algorithms are another common technique for solving capacitated facility location problems with sole sourcing. These techniques decompose a large or difficult problem into a series of easier problems. Recent research has unified the two predominant decomposition approaches, Benders decomposition and Lagrangean relaxation, into a decomposition framework called cross decomposition. Computational experience has shown that cross decomposition can efficiently solve the capacitated facility location problem. This research does not, however, provide computational experience with the capacitated facility location problem with sole sourcing.

This thesis implements two cross decomposition algorithms for the capacitated facility location problem with sole sourcing and compares these decomposition algorithms with branch and bound methods. These algorithms are implemented using a commercial solver on a high speed digital computer. For some problems tested, cross decomposition obtains better solutions in less time; however, cross decomposition does not always perform better than branch and bound due to the time required to obtain the cross decomposition bound that is theoretically superior to other decompositon bounds. The favorable results obtained using cross decomposition indicate that the method is worthy of further research.

TABLE OF CONTENTS

I. INTRODUCTION

This thesis implements two cross decomposition algorithms [Holmberg 1994] for a capacitated facility location problem with sole sourcing, compares these solution methods with direct solution using branch and bound, and applies a decomposition algorithm to a unit stationing model [Dell *et. al.* 1994]. This research evaluates the ability of a cross decomposition algorithm to determine lower bounds for the optimal objective function value of the capacitated facility location problem with sole sourcing and recommends a method to obtain feasible solutions from the cross decomposition algorithm.

The remainder of this chapter introduces the stationing model, the cross decomposition algorithm and the research approach. Chapter II describes the capacitated facility location problem and the cross decomposition method. Chapter III describes the application of two cross decomposition algorithms to the capacitated facility location problem with sole sourcing. Chapter IV presents and analyzes the test results. Chapter V offers conclusions and recommendations for further study.

A. BACKGROUND

1. Optimal Stationing of Units at Bases (OSUB)

The Defense Base Closure and Realignment Act of 1990 [BRAC 1990] specifies a process by which major Department of Defense installations can be closed or realigned. This law and subsequent legislative amendments require that installations be evaluated primarily by military value and cost. To assist the Army with stationing decisions, the OSUB (Optimal Stationing of Units to Bases) model [Dell *et. al.* 1994] was developed at the Naval Postgraduate School. This model is an elastic mixed integer programming model with two criteria: military value and cost. OSUB has recommended closure and realignment for Army maneuver and training bases and stationing Army units. OSUB is currently implemented with the General Algebraic Modeling System, GAMS, [Brooke *et. al.* 1992] and solved using the integer linear program solver XA [Sunset Technology 1987]. OSUB is a capacitated facility location problem with sole sourcing and some special constraints. The OSUB model formulation is presented below.

Indices:

* *i,i'* bases;

* *j* units;

* *k* resources (includes total maneuver acres, contiguous maneuver acres (*ca*), housing, facilities and ranges).

Data:

* S_i set of units that are currently stationed at base *i*;

* $futil_i$ fixed utility of base *i*;

* $fcost_i$ fixed cost associated with keeping base *i* open;

* $vutil_{i,j}$ the difference in variable utility when unit *j* moves to base *i* (a positive difference is a desirable change);

* $vcost_{i,j}$ the difference in variable cost when unit *j* moves to base *i* (a positive difference is a higher cost);

* $pen_{i,k}$ penalty per unit deviation from resource *k* at base *i* (any deviation not associated with a military value objective has a penalty of zero);

* $co_{i,k}$ operating cost associated with deviating from resource *k* at base *i* (any deviation not associated with the cost objective has a cost of zero);

* $cc_{i,k}$ construction cost associated with deviating from resource *k* at base *i* (any deviation not associated with a construction cost has a cost of zero);

* $cap_{i,k}$ capacity of resource *k* at base *i* (current stationed unit use is subtracted from the capacity for all $k \neq ca$);

* $r_{j,k}$ resource *k* utilization by unit *j*;

* $cm_{i,j}$ cost to move unit *j* to base *i*;

* *maxm* the maximum movement cost;

* *maxc* the maximum one-time realignment cost.

Binary Variables:

- $close_i = 1$ if base i is closed and 0 if it remains open;

- $move_{i,j} = 1$ if unit j moves to base i. (This variable is defined only for units j not already assigned to base i.)

Continuous Variables:

- $e_{i,k}$ deviation from resource capacity k at base i.

Formulation:

(1) maximize $Z_1 = \sum_i futil_i \cdot (1 - close_i) + \sum_i \sum_{j \notin S_i} vutil_{i,j} \cdot move_{i,j} - \sum_i \sum_k pen_{i,k} \cdot e_{i,k}$

(2) minimize $Z_2 = \sum_i fcost_i \cdot (1 - close_i) + \sum_i \sum_{j \notin S_i} vcost_{i,j} \cdot move_{i,j} + \sum_i \sum_k co_{i,k} \cdot e_{i,k}$

subject to:

(3) $\sum_{i' \neq i} move_{i',j} \leq 1 \quad \forall i, j \in S_i$

(4) $move_{i,j} \leq (1 - close_i) \quad \forall i, j \notin S_i$

(5) $\sum_{i' \neq i} move_{i',j} \geq close_i \quad \forall i, j \in S_i$

(6) $\sum_{j \notin S_i} r_{j,k} \cdot move_{i,j} - \sum_{j \in S_i} \sum_{i' \neq i} r_{j,k} \cdot move_{i',j} \leq cap_{i,k} + e_{i,k} \quad \forall i, k \neq ca$

(7) $r_{j,k} \cdot move_{i,j} \leq cap_{i,k} + e_{i,k} \quad \forall i, j \notin S_i, k = ca$

(8) $\sum_i \sum_j cm_{i,j} \cdot move_{i,j} \leq maxm$

(9) $\sum_i \sum_k cc_{i,k} \cdot e_{i,k} \leq maxc$

(10) $\mathrm{move}_{i,j} \in \{0,1\} \ \forall i, j \notin S_i \quad close_i \in \{0,1\} \ \forall i \quad \mathrm{e}_{i,k} \geq 0 \ \forall i, k$

These objectives and constraints respectively:

(1) express a comparative measure of military value for units assigned to bases,

(2) express the cost of unit stationing,

(3) ensure a unit moves at most once or not at all,

(4) & (5) ensure that a base closes only if no new units are stationed there and all old units move to another base,

(6) capacitate housing, total maneuver acres, facilities and ranges or measure excess demand for such,

(7) capacitate contiguous maneuver acres or measure excess demand for such,

(8) limit the maximum unit movement cost, and

(9) limit the one-time construction cost for realignment.

2. Cross Decomposition

Van Roy [1983, 1986] develops a cross decomposition algorithm that unifies Benders decomposition and Lagrangean relaxation and applies it to the capacitated facility location problem. This method simultaneously isolates and exploits the primal and the dual structure of the problem by successively solving a transportation problem and a simple plant location problem. Van Roy presents evidence that this algorithm is superior to several other methods and shows that an implementation of the algorithm solves sample problems about ten times faster than other methods available at that time. Van Roy also claims that in all problems tested his method obtains in just a few decomposition iterations tight lower and upper bounds that differ by no more than 0.5%.

Holmberg [1990, 1994] generalizes the concept of cross decomposition to pure integer problems and studies the lower bounds on the optimal objective function value of pure integer programming problems. He proves that generalized Benders decomposition and generalized cross decomposition yield the best lower bound for pure integer problems. He also shows that while either ordinary Benders decomposition or Lagrangean relaxation may yield the best lower bound for a particular problem, cross decomposition can automatically yield the best of these bounds. He formulates the decomposition problems required to apply generalized cross decomposition to a capacitated facility location problem with sole sourcing. He provides no computational evidence of the efficiency of these methods.

B. OVERVIEW

This thesis presents three accomplishments.

1. Implementing Cross Decomposition

Two cross decomposition algorithms are implemented for the capacitated facility location problem with sole sourcing. The algorithms are coded in C on an IBM System 6000 Model 590H using the CPLEX Callable Library. CPLEX [1994] is an optimization tool for solving linear and mixed integer optimization problems. The CPLEX Callable Library is an object-oriented C library. CPLEX allows the user to build applications which solve, modify, and interpret the results of linear and mixed integer programs. To code the algorithms we develop techniques to successively generate and solve the decomposition problems.

2. Analysis of Results

The cross decomposition algorithms are tested and analyzed. We perform computational tests using the decomposition algorithms and the CPLEX Mixed Integer Solver on problems of various sizes. We compare the results and characterize the performance of the decomposition algorithms. This analysis evaluates the capability of the cross decomposition algorithms to determine lower bounds on the optimal objective function value for capacitated facility location problems with sole sourcing. We also analyze the suitability of the algorithms to obtain feasible solutions using the facility configurations and lower bounds from the decomposition algorithms.

3. Application to Unit Stationing Models

A decomposition approach is selected and used on a unit stationing model similar to the OSUB model. The unit stationing model is simple extension of the capacitated facility location problem with sole sourcing.

II. CAPACITATED FACILITY LOCATION AND CROSS DECOMPOSITION

A. CAPACITATED FACILITY LOCATION PROBLEMS

The CFLP describes a wide variety of planning problems. Applications beyond capacitated facility location include: lot sizing decisions in production planning; telecommunications network design; machine replacement; vehicle routing when capacities are not equal [Cornuejols *et. al.* 1991]; the stochastic transportation problem; and discrete network design [Holmberg 1990]. The mixed integer formulation of the CFLP is presented below.

Indices:

* i facilities;
* j demand points.

Data:

* f_i fixed cost to operate facility i;
* $c_{i,j}$ cost to supply all demand at j by facility i;
* d_j total demand at j.

Binary Variables:

* $y_i = 1$ if facility i is open and 0 if it is closed.

Continuous Variables:

* $x_{i,j}$ the proportion of the total demand at j (d_j) provided by facility i.

Formulation:

$$\text{(CFLP)} \quad \underset{y,x}{\text{minimize}} \quad \sum_{i} f_i \cdot y_i + \sum_{i,j} c_{i,j} \cdot x_{i,j}$$

subject to:

$$\text{(CFLP 1)} \quad \sum_{i} x_{i,j} = 1 \quad [\lambda_j] \quad \forall j$$

$$\text{(CFLP 2)} \quad \sum_{j} d_j \cdot x_{i,j} \le s_i \cdot y_i \quad [\mu_i] \quad \forall i$$

$$\text{(CFLP 3)} \quad x_{i,j} \le y_i \quad [\nu_{i,j}] \quad \forall i,j$$

$$\text{(CFLP 4)} \quad x_{i,j} \ge 0 \quad \forall i,j \qquad y_i \in \{0,1\} \quad \forall i$$

The objective is to minimize total cost, and contains two distinct sets of decision variables. The first set of binary decision variables (y_i) determine which facilities to open. The second set of continuous decision variables ($x_{i,j}$) allocate customer demand to the open facilities. The variables in brackets are the corresponding dual variables for the given set of constraints. Constraints (CFLP 1) ensure that all demand is met. Constraints (CFLP 2) enforce the capacity limits for the facilities. Constraints (CFLP 3) are variable upper bounds on the allocation of demand. This third set of constraints is redundant in binary variables, but yields a much tighter linear programming relaxation [Van Roy 1986].

B. SOLVING THE CFLP

1. Benders Decomposition

Benders decomposition (*e.g.*, [Nemhauser & Wolsey 1988] and [Magnanti & Wong 1990]) is a primal solution method. It isolates special structure in the problem by fixing primal variables. Benders decomposition is an exact method that solves the CFLP optimally by iterating between the primal subproblem and the primal master problem described below. Appendix A contains a derivation of the master problem and subproblem. The description below provides implementation details.

a. *The Primal Subproblem (PS)*

$$\text{(PS)}\quad \underset{x}{\text{minimize}} \quad \sum_i f_i \cdot \hat{y}_i + \sum_{i,j} c_{i,j} \cdot x_{i,j}$$

subject to:

$$\text{(PS 1)}\quad \sum_i x_{i,j} = 1 \quad [\lambda_j] \quad \forall j$$

$$\text{(PS 2)}\quad \sum_j d_j \cdot x_{i,j} \le s_i \cdot \hat{y}_i \quad [\mu_i] \quad \forall i$$

$$\text{(PS 3)}\quad x_{i,j} \le \hat{y}_i \quad [\nu_{i,j}] \quad \forall i,j$$

$$\text{(PS 4)}\quad x_{i,j} \ge 0 \quad \forall i,j$$

The primal subproblem is the linear program obtained when the facility configuration is fixed at $\hat{y}$ in the CFLP where $\hat{y}_i$ indicates that y_i is fixed at either one or zero. PS is a restriction of the CFLP that consequently provides an upper bound on its optimal objective function value. The objective function value of the dual of PS is

$$\sum_j \lambda_j - \sum_i s_i \cdot \hat{y}_i \cdot \mu_i - \sum_{i,j} \hat{y}_i \cdot \nu_{i,j} .$$

To maintain consistency with the objective function of the CFLP, we add the same constant,

$$\sum_i f_i \cdot \hat{y}_i$$

(the fixed facility cost), that is added to PS and obtain

$$\sum_i f_i \cdot \hat{y}_i + \sum_j \lambda_j - \sum_i s_i \cdot \hat{y}_i \cdot \mu_i - \sum_{i,j} \hat{y}_i \cdot \nu_{i,j} .$$

By duality, the value of the above expression for the optimal solution to PS is the maximum feasible value for the given facility configuration. This expression provides the basis for a primal cut. The optimal objective function value of the primal master problem must be less than or equal to

$$\sum_j \lambda_j^{(t)} + \sum_i (f_i - \mu_i^{(t)} \cdot s_i - \sum_j \nu_{i,j}^{(t)}) \cdot y_i$$

for any set of feasible facility configurations (where the superscript (*t*) on the dual variables is the iteration number).

b. The Primal Master Problem (PM)

(PM) $$\underset{y,\rho}{\text{minimize}} \quad \rho$$

subject to:

(PM 1) $$\rho \geq \sum_j \lambda_j^{(t)} + \sum_i (f_i - \mu_i^{(t)} \cdot s_i - \sum_j \nu_{i,j}^{(t)}) \cdot y_i \quad \forall t$$

(PM 2) $$\sum_i s_i \cdot y_i \geq \sum_j d_j$$

(PM 3) $$y_i \in \{0,1\} \quad \forall i$$

The primal master problem is obtained by adding a primal cut (PM 1) each time the primal subproblem is solved. PM fixes the facility location variables for the primal subproblem. PM provides a feasible facility configuration because constraint (PM 2) ensures total supply meets total demand.

When using only a subset of all possible cuts (a relaxation), PM provides a lower bound on the optimal objective function value of the CFLP. Since PS determines the maximum value of

$$\sum_j \lambda_j^{(t)} + \sum_i (f_i - \mu_i^{(t)} \cdot s_i - \sum_j \nu_{i,j}^{(t)}) \cdot y_i$$

for the facility configuration provided by PM, the optimal objective function value of the CFLP is identified when the objective function value of PS equals the objective function value of PM.

2. Lagrangean Relaxation

Lagrangean relaxation [e.g., Nemhauser and Wolsey 1988] is a dual solution method. It isolates special structure in the problem by moving complicating constraints into the objective function and penalizing complicating constraint infeasibility with a Lagrangean multiplier. Lagrangean relaxation solves a relaxation of the CFLP by iterating between the subproblem and the master problem described below.

a. *The Dual Subproblem (DS)*

(DS) $$\underset{x,y}{\text{minimize}} \quad \sum_i f_i \cdot y_i + \sum_{i,j} c_{i,j} \cdot x_{i,j} + \sum_i \left(\sum_j d_j \cdot x_{i,j} - s_i \cdot y_i \right) \cdot \hat{\mu}_i$$

subject to:

(DS 1) $$\sum_i x_{i,j} = 1 \quad \forall j$$

(DS 2) $$\sum_i s_i \cdot y_i \geq \sum_j d_j$$

(DS 3) $$x_{i,j} \leq y_i \quad \forall i,j$$

(DS 4) $$x_{i,j} \geq 0 \quad \forall i,j \qquad y_i \in \{0,1\} \quad \forall i$$

The dual subproblem is the Lagrangean relaxation of the CFLP with respect to the total supply constraint (CFLP 2) and the addition of a constraint (DS 2) that ensures total supply meets total demand. DS is a relaxation of the CFLP that consequently provides a lower bound on its optimal objective function value. DS provides for any set of dual values fixed facility locations (y_i) and customer assignments ($x_{i,j}$).

b. The Dual Master Problem (DM)

(DM)
$$\underset{\mu,\delta}{\text{maximize}} \quad \delta$$

subject to:

(DM 1)
$$\delta \le \sum_i f_i \cdot y_i^{(t)} + \sum_{i,j} c_{i,j} \cdot x_{i,j}^{(t)} + \sum_i \left(\sum_j d_j \cdot x_{i,j}^{(t)} - s_i \cdot y_i^{(t)} \right) \cdot \mu_i \quad \forall t$$

(DM 2)
$$\mu_i \ge 0 \quad \forall i$$

The dual master problem is obtained by adding a dual cut (DM 1) each time the dual subproblem is solved. With all possible cuts, DM is the Lagrangean dual. DM fixes the dual variables for the dual subproblem. DM provides an upper bound on the optimal objective function value of the Lagrangean relaxation of the CFLP because it is a relaxation of the Lagrangean dual.

3. Cross Decomposition

Cross decomposition unifies Benders decomposition and Lagrangean relaxation. Figure 1 below motivates the cross decomposition method. Cross decomposition iterates in the subproblem phase between the restricted primal subproblem and the relaxed dual subproblem described above. These problems successively provide an upper and lower bound on the optimal objective function value of the CFLP. These problems may provide tight bounds in the subproblem phase, but neither convergence, nor monotonic improvement is guaranteed. The convergence tests described below are used to determine when the subproblems fail to make progress toward an optimal solution. When a convergence test fails, cross decomposition solves a master problem that is formed using the cuts generated in the subproblem phase and then continues this phase with the next subproblem.

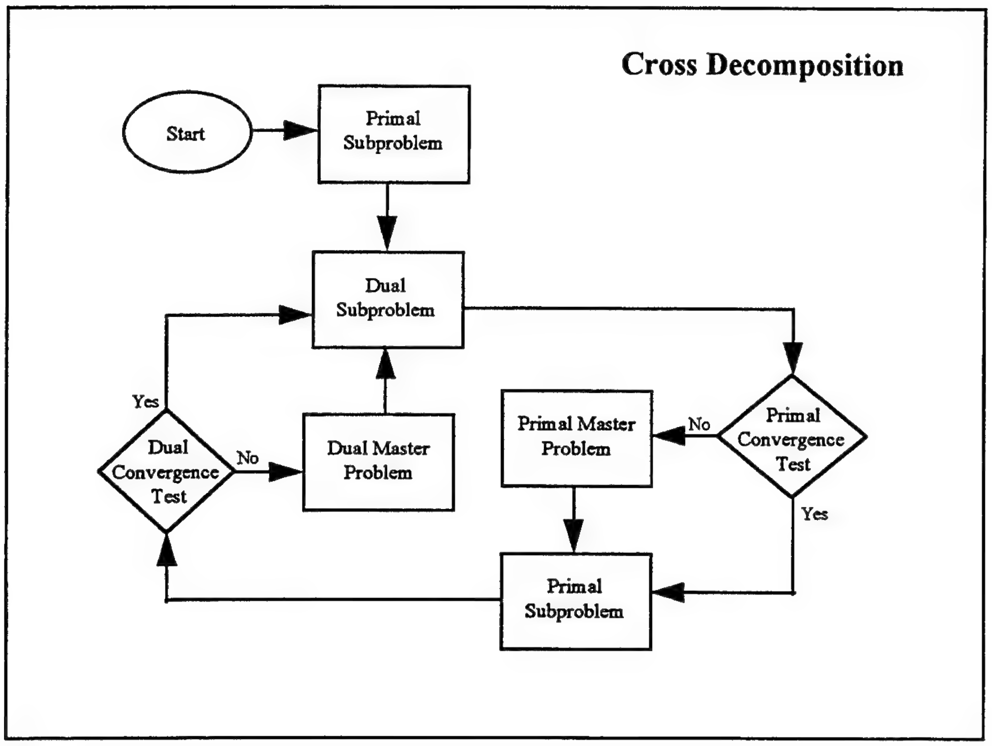

Figure 1.

Cross Decomposition iterates between the primal and dual subproblems in the "subproblem phase." It solves a primal or dual master problem when a convergence test fails and then restarts the subproblem phase. The algorithm terminates when the objective function values of selected problems converge to a lower bound on the optimal objective function value.

a. *The Primal Convergence Test*

The primal convergence test uses the following cuts:

$$\sum_j \lambda_j^{(t)} + \sum_i \left\{ f_i - \mu_i^{(t)} \cdot s_i - \sum_j \nu_{i,j}^{(t)} \right\} \cdot \hat{y}_i \leq \textit{Upper Bound}$$

If these cuts are satisfied for all *t* then cross decomposition continues by solving the primal subproblem. The primal convergence test uses the cuts in the primal master problem to determine if the upper bound can be improved. If any cut *t* is not satisfied, a master problem is solved. The primal convergence test is a necessary condition for improving the current upper bound on the optimal objective function value. The primal convergence test is sufficient to show that either: the primal subproblem can improve the upper bound; or the primal subproblem can generate a new cut for the primal master problem [Holmberg 1990].

b. The Dual Convergence Test

The dual convergence test uses the following cuts:

$$\sum_{i,j} c_{i,j} \cdot x_{i,j}^{(t)} + \sum_{i} f_i \cdot y_i^{(t)} + \sum_{i} \left\{ \sum_{j} d_j \cdot x_{i,j}^{(t)} - s_i \cdot y_i^{(t)} \right\} \cdot \hat{\mu}_i \geq Lower\ Bound \quad .$$

If these cuts are all satisfied then cross decomposition continues by solving the dual subproblem. The dual convergence test uses the cuts in the dual master problem to determine if the lower bound can be improved. If any of these cuts are not satisfied, cross decomposition solves a master problem. The dual convergence test is a necessary condition for improving the current lower bound on the optimal objective function. The dual convergence test is sufficient to show that either: the dual subproblem can improve the lower bound; or the dual subproblem can generate a new cut for the dual master problem [Holmberg 1990].

III. CAPACITATED FACILITY LOCATION WITH SOLE SOURCING

The capacitated facility location problem with sole sourcing (CFLPSS) is an important variant of the CFLP model. In this problem all customer demand must be assigned to one facility and the second set of decision variables ($x_{i,j}$) are also binary. The CFLPSS is a pure integer linear program with significantly more binary variables than the related CFLP. To apply cross decomposition to CFLPSS we first reformulate the problem as follows:

$$\text{(SS)} \quad \underset{y,x}{\text{minimize}} \quad \sum_i f_i \cdot y + \sum_{i,j} c_{i,j} \cdot x_{i,j}$$

subject to:

$$\text{(SS 1)} \quad W: \begin{cases} \sum_i x_{i,j} = 1 & [\lambda_j] \quad \forall j \\ x_{i,j} \le y_i & [\nu_{i,j}] \quad \forall i,j \end{cases}$$

$$\text{(SS 2)} \quad X_i: \begin{cases} \sum_j d_j \cdot x_{i,j} \le s_i & [\omega_i] \\ x_{i,j} \in \{0,1\} \quad \forall j \end{cases} \quad \forall i$$

$$\text{(SS 3)} \quad Y: \begin{cases} \sum_i s_i \cdot y_i \ge \sum_j d_j & [\theta] \\ y_i \in \{0,1\} \quad \forall i \end{cases}$$

This formulation partitions the constraints into three sets: those that include only the facility location decisions (*Y*), those that include only customer assignment decisions (X_i), and those that involve both types of decisions (*W*). Cross decomposition exploits this structure. Note that the constraint in CFLP that limits the assignment of customer demand to the available supply of open facilities (CFLP 2) has been replaced in CFLPSS by a constraint in X_i that limits customer demand to the available supply of the facility. Thus the variable upper bounds (now in *W)* are no longer redundant.

A. GCD APPLIED TO CFLPSS (CX)

Holmberg [1994] describes the problems and the general procedures to apply cross decomposition to CFLPSS. We adopt his notation with only minor modifications to label and classify the various decomposition problems. In this section we formulate the problems associated with generalized cross decomposition (GCD) and describe our implementation of GCD. We highlight where appropriate the implementation details not provided by Holmberg [1990, 1994]. In the next section we formulate the problems associated with one of the six alternate ordinary cross decomposition algorithms (CD6) described but not implemented by Holmberg and describe our implementation of that decomposition.

1. Lagrangean Relaxation

Our implementation of Lagrangean Relaxation (DW3) iterates between the subproblem and master problem described below. (Holmberg [1994] describes the subproblems and master problems for three possible dual decomposition algorithms that he labels Danzig-Wolfe decomposition.) When applied to CFLPSS this algorithm converges to the Lagrangean relaxation of the constraints in the set *W*.

a. *The Dual Subproblem (SSDS)*

$$\text{(SSDS)} \qquad \underset{y,x}{\text{maximize}} \sum_i g_i^{(x)}(\lambda, \nu) + g^{(y)}(\nu) - \sum_j \lambda_j$$

Where for fixed dual values we solve the following subproblems:

$$\text{(SSDS 1)}\ DSY: \begin{cases} g^{(y)}(\hat{\nu}) = \underset{y}{\text{minimize}} \sum_i \left\{ f_i - \sum_j \hat{\nu}_{i,j} \right\} \cdot y_i \\ \\ \text{subject to:} \begin{cases} \sum_i s_i \cdot y_i \geq \sum_j d_j \\ y_i \in \{0,1\} \quad \forall i \end{cases} \end{cases}$$

$$\text{(SSDS 2)}\quad DSX_i: \left\{ \begin{array}{l} g_i^{(x)}(\hat{\lambda},\hat{\nu}) = \begin{array}{c}\text{minimize} \\ x \in X_i\end{array} \sum_j (c_{i,j} + \hat{\lambda}_j + \hat{\nu}_{i,j}) \cdot x_{i,j} \\ \\ \qquad \text{subject to: } \left\{ \begin{array}{l} \sum_j d_j \cdot x_{i,j} \le s_i \\ x_{i,j} \in \{0,1\} \quad \forall j \end{array} \right. \end{array} \right\} \quad \forall i$$

The dual subproblem SSDS is the Lagrangean relaxation of the CFLPSS relative to the constraints in *W*. This relaxation creates a set of independent binary knapsack problems. DSY selects facilities to open. DSX_i assigns customers to facility *i* without regard to either the open facilities selected by DSY or the assignments made to other facilities. This relaxed problem provides a lower bound and a set of cuts for the master problem.

b. The Dual Master Problem (SSDM)

$$\text{(SSDM)}\quad \begin{array}{c}\text{maximize} \\ \nu,\lambda\end{array} \sum_i \delta_i^{(x)} + \delta^{(y)} - \sum_j \lambda_j$$

subject to:

$$\text{(SSDM 1)}\quad \delta_i^{(x)} \le \sum_j (x_{i,j}^{(t_x)} \cdot c_{i,j} + x_{i,j}^{(t_x)} \cdot \lambda_j + x_{i,j}^{(t_x)} \cdot \nu_{i,j}) \quad \forall i, t_x$$

$$\text{(SSDM 2)}\quad \delta^{(y)} \le \sum_i f_i \cdot y_i^{(t_y)} - \sum_{i,j} y_i^{(t_y)} \nu_{i,j} \quad \forall t_y$$

$$\text{(SSDM 3)}\quad \nu_{i,j} \ge 0 \quad \forall i,j$$

The dual master problem SSDM is obtained by adding dual cuts (t_x and t_y) each time the respective dual subproblem is solved. This problem provides an upper bound and dual values for the subproblem. To ensure that SSDM is initially bounded, we include a set of initial cuts corresponding to all facilities open and all customers assigned to each facility. Holmberg [1990, 1994] does not suggest this.

2. Generalized Benders Decomposition

Generalized Benders decomposition (GBD) iterates between the master problem and subproblem described below. When applied to CFLPSS, it converges to the convexification with respect to the assignment variables (CX). (The convexification with respect to a set of integer variables is the optimal solution over the feasible region defined by the convex hull of the integer feasible extreme points in the set.) This is the best lower bound that we can obtain using decomposition [Holmberg 1994].

a. *The Primal Subproblem (PSL)*

(PSL) $$\underset{x}{\text{minimize}} \quad g_i^{(x)}(\lambda, \nu) - \sum_j \lambda_j + \sum_i f_i \cdot \hat{y}_i$$

subject to:

(PSL 1) $\nu_{i,j} \geq 0 \quad \forall i, j$

Where for fixed dual values we solve:

(PSL 2) $$\text{PSDS}_i : \left\{ \begin{array}{l} g_i^{(x)}(\hat{\lambda}, \hat{\nu}) = \underset{x \in X_i}{\text{minimize}} \displaystyle\sum_j (c_{i,j} + \hat{\lambda}_j + \hat{\nu}_{i,j}) \cdot x_{i,j} \\ \\ \qquad \text{subject to:} \left\{ \begin{array}{l} \displaystyle\sum_j d_j \cdot x_{i,j} \leq s_i \\ x_{i,j} \in \{0,1\} \quad \forall j \end{array} \right. \end{array} \right\} \quad \forall i$$

The primal subproblem, PSL, is the Lagrangean relaxation of CFLPSS with respect to the constraints *W* after the facility locations have been fixed. Since this problem is a nonlinear mixed integer program, we do not solve it directly. It may be solved approximately using subgradient optimization, or optimally using Danzig-Wolfe decomposition. The master and subproblems used to solve PSL optimally are given below followed by brief description of the solution method.

(1) Danzig-Wolfe Master Problem to solve PSL (DML)

$$\text{(DML)} \quad \underset{\nu,\lambda}{\text{maximize}} \quad \sum_i \delta_i^{(x)} + \delta^{(\hat{y})} - \sum_j \lambda_j$$

subject to:

$$\text{(DML 1)} \quad \delta^{(\hat{y})} \le \sum_i f_i \cdot \hat{y}_i - \sum_{i,j} \hat{y}_i \cdot \nu_{i,j}$$

$$\text{(DML 2)} \quad \delta_i^{(x)} \le \sum_j (x_{i,j}^{(k)} \cdot c_{i,j} + x_{i,j}^{(k)} \cdot \lambda_j + x_{i,j}^{(k)} \cdot \nu_{i,j}) \quad \forall i,k$$

$$\text{(DML 3)} \quad \nu_{i,j} \ge 0 \quad \forall i,j$$

(2) Danzig-Wolfe Subproblem to Solve PSL (PSDS$_i$)

$$\text{PSDS}_i: \left\{ \begin{array}{ll} g_i^{(x)}(\hat{\lambda},\hat{\nu}) = & \underset{x \in X_i}{\text{minimize}} \ \sum_j (c_{i,j} + \hat{\lambda}_j + \hat{\nu}_{i,j}) \cdot x_{i,j} \\ & \text{subject to:} \left\{ \begin{array}{l} \sum_j d_j \cdot x_{i,j} \le s_i \\ x_{i,j} \in \{0,1\} \quad \forall j \end{array} \right. \end{array} \right\} \quad \forall i$$

b. Solving PSL

To solve PSL, we iterate between the Danzig-Wolfe subproblem, PSDS$_i$, and the Danzig-Wolfe master problem, DML. PSDS$_i$ is identical to DSX$_i$ from the dual subproblem (SSDM). DML is SSDM with the *y* cuts (SSDM 2) aggregated into a single constraint (DML 1). Holmberg [1994] provides these problems, but not the implementation details that follow in this section. These problems converge when

$$g_i^{(x)}(\hat{\lambda},\hat{\nu}) \ge \delta_i^{(x)} \quad \forall i$$

To synthesize dual information for closed facilities, we use the following rule.

$$\text{If } \hat{y}_i = 0 \text{ then } \hat{v}_{i,j} = \begin{cases} 0 & \text{if } c_{i,j} + \hat{\lambda}_j \geq 0 \\ \left| c_{i,j} + \hat{\lambda}_j \right| & \text{otherwise} \end{cases}$$

This rule ensures that (DML 2) is satisfied for the closed facilities. When DML is unbounded the facility choices are infeasible and we add the following constraint to PMCX and SSDS:

$$\sum_{i \notin I} y_i \geq 1 \text{ where } I = \{ i \mid \hat{y}_i = 1 \}.$$

Holmberg assumes feasibility for CFLPSS when the constraints are partitioned into the set X and the set Y. If this assumption does not hold the above constraint is required to achieve feasibility. We retain the cuts in DML that were found each time PSL was solved since these are valid at every iteration between PSL and PMCX (below).

c. *The Primal Master Problem (PMCX)*

(PMCX)
$$\underset{\rho, y}{\text{minimize}} \quad \rho$$

subject to:

(PMCX 1)
$$\rho \geq \sum_{i,j} (c_{i,j} + \lambda_j^{(t)} + v_{i,j}^{(t)}) \cdot x_{i,j}^{(t)} - \sum_j \lambda_j^{(t)} + \sum_i (f_i - \sum_j v_{i,j}^{(t)}) \cdot y_i \quad \forall t$$

(PMCX 2)
$$y_i \in Y: \begin{cases} \sum_i s_i \cdot y_i \geq \sum_j d_j \\ y_i \in \{0,1\} \quad \forall i \end{cases}$$

The primal master problem PMCX is the convexification with respect to the assignment variables ($x_{i,j}$). It is formed by adding a primal cut each time the primal subproblem is solved. It contains constraint (PMCX 2) to ensure that enough facilities are open to meet total demand. This is a necessary condition for primal feasibility. This is not a sufficient condition for primal feasibility; therefore, additional constraints are required. We detect infeasibility when DML is unbounded in PSL and add a constraint (described in section **b.** above) to achieve feasibility.

3. Generalized Cross Decomposition

Generalized cross decomposition (GCD) iterates between PSL and SSDS until a convergence test fails or these problems converge in objective value. GCD converges to the convexification with respect to the assignment variables (CX). This is the better of the two lower bounds provided by DW3 and GBD if they differ. If the GBD lower bound exceeds the DW3 upper bound, we switch to GBD to find the better lower bound. The convergence tests for GCD are described below.

a. The Primal Convergence Test (CTPCX)

CTPCX uses the following cuts:

$$\sum_{i,j}\left\{c_{i,j}+\lambda_j^{(t)}+v_{i,j}^{(t)}\right\}\cdot x_{i,j}^{(t)}-\sum_j \lambda_j^{(t)}+\sum_i\left\{f_i-\sum_j v_{i,j}^{(t)}\right\}\cdot \hat{y}_i \le \textit{Upper Bound}.$$

If all of these cuts are satisfied then GCD continues by solving PSL. CTPCX uses the cuts in PMCX to determine if PSL can improve the upper bound. If any cut is not satisfied we solve PMCX. CTPCX also fails if PMCX has not been solved for a fixed number of iterations.

b. The Dual Convergence Test (SSCTD)

SSCTD uses the following cuts:

$$\sum_{i,j} c_{i,j}\cdot x_{i,j}^{(t_x)}+\sum_i f_i\cdot y_i^{(t_y)}+\sum_{i,j}\left\{x_{i,j}^{(t_x)}-y_i^{(t_y)}\right\}\cdot \hat{v}_{i,j}+\sum_j\left\{\sum_i x_{i,j}^{(t_x)}-1\right\}\cdot \hat{\lambda}_j \ge \textit{Lower Bound}.$$

If all cuts are satisfied then GCD continues by solving SSDS. SSCTD uses the cuts in SSDM to determine if SSDS can improve the lower bound. If any cut is not satisfied we solve SSDM.

B. CROSS DECOMPOSITION APPLIED TO CFLPSS (LX)

We can relax the integer restrictions on the customer assignment variables and obtain another decomposition that converges to the same objective value as the linear programming relaxation of CFLPSS with respect to the assignment variables. This form of decomposition is similar to the decomposition described by Van Roy [1986]. In this form we replace the primal master problem PMCX with PMLX and the primal subproblem PSL with PSLX. We also replace the primal convergence test CTPCX with CTPLX. The dual problems and the dual convergence test remain unchanged. These new primal problems and primal convergence test are detailed below.

1. Benders Decomposition on LX

Benders decomposition on LX (BD6) iterates between the following subproblem and master problem and converges to the linear programming relaxation of the assignment variables (LX).

a. The Primal Subproblem (PSLX)

$$\text{(PSLX)}\quad \underset{x}{\text{minimize}} \quad \sum_i f_i \cdot \hat{y}_i + \sum_{i,j} c_{i,j} \cdot x_{i,j}$$

subject to:

$$\text{(PSLX 1)}\quad W: \begin{cases} \sum_i x_{i,j} = 1 & [\lambda_j] \quad \forall j \\ x_{i,j} \le \hat{y}_i & [\nu_{i,j}] \quad \forall i,j \end{cases}$$

$$\text{(PSLX 2)}\quad X_i: \begin{cases} \sum_j d_j \cdot x_{i,j} \le s_i & [\omega_i] \\ x_{i,j} \ge 0 \quad \forall j \end{cases} \quad \forall i$$

The primal subproblem (PSLX) is the linear programming relaxation of the original problem (CFLPSS) with respect to the assignment variables after the facility locations have been fixed. It provides an upper bound and a set of dual values to form the cut for the master problem (below).

b. *The Primal Master Problem (PMLX)*

$$\text{(PMLX)} \quad \underset{y}{\text{minimize}} \quad \rho$$

subject to:

$$\text{(PMLX 1)} \quad \rho \geq \sum_j \lambda_j^{(t)} - \sum_i s_i \cdot \omega_i^{(t)} + \sum_i \left\{ f_i - \sum_j v_{i,j}^{(t)} \right\} \cdot y_i \qquad \forall t$$

$$\text{(PMLX 2)} \quad Y: \begin{cases} \sum_i s_i \cdot y_i \geq \sum_j d_j \\ y_i \in \{0,1\} \quad \forall i \end{cases}$$

The primal master problem is obtained by adding a cut (PMLX 1) each time the PSLX is solved. It fixes the facility location variables for the next decomposition problem, and provides a lower bound.

2. Cross Decomposition on LX

Cross decomposition on LX (CD6) iterates between PSLX and SSDS in the subproblem phase until a convergence test fails or these problems converge in objective value. The new convergence test for GCD and procedures for objective convergence are described below.

a. *The Primal Convergence Test (CTPLX)*

CTPLX uses the following cuts:

$$\sum_j \lambda_j^{(t)} - \sum_i s_i \cdot \omega_i^{(t)} + \sum_i \left\{ f_i - \sum_j v_{i,j}^{(t)} \right\} \cdot \hat{y}_i \leq \textit{Upper Bound}$$

If these cuts are all satisfied then CD6 continues by solving PSLX. CTPLX uses the cuts in PMLX to determine if the upper bound can be improved. PMLX is solved if any cut is not satisfied or if PMLX has not been solved for a fixed number of iterations.

b. Convergence to a Bound

CD6 can converge to the better of the bounds provided by BD6 and DW3, even if it is not known which is better in advance. If during cross decomposition the current BD6 lower bound exceeds the current DW3 upper bound, then cross decomposition switches to BD6. Likewise, if the current DW3 lower bound exceeds the current BD6 upper bound, then the algorithm switches to DW3.

C. DECOMPOSITION APPLIED TO A STATIONING MODEL

In this section we describe the application of a decomposition algorithm to solve a simplified unit stationing model. That model is given below.

$$\text{(USM)} \quad \text{minimize} \quad \sum_i f_i \cdot y_i + \sum_{i,j} c_{i,j} \cdot x_{i,j}$$

subject to:

$$\text{(USM 1)} \quad W: \begin{cases} \sum_i x_{i,j} = 1 \quad [\lambda_j] \quad \forall j \\ x_{i,j} \le y_i \quad [v_{i,j}] \quad \forall i,j \end{cases}$$

$$\text{(USM 2)} \quad X_i: \begin{cases} \sum_j d_{j,k} \cdot x_{i,j} \le s_{i,k} \quad [\omega_{i,k}] \quad \forall k \in R_0 \\ d_{j,k} \cdot x_{i,j} \le s_{i,k} \quad [\eta_{i,j,k}] \quad \forall j,k \in R_1 \\ x_{i,j} \in \{0,1\} \quad \forall j \end{cases} \qquad \forall i$$

$$\text{(USM 3)} \quad Y: \begin{cases} \sum_i s_{i,k} \cdot y_i \ge \sum_j d_{j,k} \quad [\theta_k] \quad \forall k \in R_0 \\ y_i \in \{0,1\} \quad \forall i \end{cases}$$

This unit stationing model (USM) is an extension of the CFLPSS model that includes multiple resources. These resources are separated into two categories. One category (R_0) is resources that are consumed by the presence of the unit such as motor pools and billets. The second category (R_1) is resources such as ranges and maneuver acres that are required by the unit, but are shared for the most part with other units. This simplified model captures the essential aspects of OSUB [Dell *et. al.* 1994]. Note that this model has the same decision variables as the CFLPSS model with the same number of customers and facilities.

We do not solve the model directly; instead we satisfy the constraints associated with the second resource category (R_1) by variable reduction. That is, we fix the assignment variable to zero for any unit and facility pair that is not feasible due to a shortage of one of these resources.

1. Benders Decomposition

Benders decomposition on LX (BDLX) for the unit stationing model iterates between the following subproblem and master problem and converges to the linear programming relaxation of the assignment variables (LX). This decomposition is similar to BD6.

a. *The Primal Subproblem (USMPS)*

$$\text{(USM PSLX)} \quad \underset{x}{\text{minimize}} \quad \sum_i f_i \cdot \hat{y}_i + \sum_{i,j} c_{i,j} \cdot x_{i,j}$$

subject to:

$$\text{(USM PSLX 1)} \quad W: \begin{cases} \sum_i x_{i,j} = 1 \quad [\lambda_j] \quad \forall j \\ x_{i,j} \le \hat{y}_i \quad [\nu_{i,j}] \quad \forall i,j \end{cases}$$

$$\text{(USM PSLX 2)} \quad X_i: \begin{cases} \sum_j d_{j,k} \cdot x_{i,j} \le s_{i,k} \quad [\omega_{i,k}] \quad \forall k \in R_0 \\ 0 \le x_{i,j} \le 1 \quad \forall j \\ x_{i,j} = 0 \ \text{ if } \exists k \in R_1 | (s_{i,k} / d_{j,k}) < 1 \end{cases} \qquad \forall i$$

The primal subproblem is the linear programming relaxation of USM after the facility location variables have been fixed. Note that the relaxed assignment variable is zero if the facility can not provide enough of a resource in the second resource category to the unit. USMPS provides an upper bound on LX and a set of dual variables that form the cut for the master problem (below).

b. The Primal Master Problem (USMPM)

(USM PMLX) $$\underset{\rho, y}{\text{minimize}} \quad \rho$$

subject to:

(USM PMLX 1) $$\rho \geq \sum_{i} f_i \cdot y_i + \sum_{j} \lambda_j^{(t)} - \sum_{i,j} v_{i,j}^{(t)} \cdot y_i - \sum_{i,k \in R_0} \omega_{i,k}^{(t)} \cdot s_{i,k} \qquad \forall t$$

(USM PMLX 2) $$Y: \begin{cases} \sum_{i} s_{i,k} \cdot y_i \geq \sum_{j} d_{j,k} & [\theta_k] \quad \forall k \in R_0 \\ y_i \in \{0,1\} \quad \forall i \end{cases}$$

The primal master problem is formed by adding a cut (USM PMLX 1) each time the primal subproblem is solved. Note that the constraints in USM PMLX 2 require that the open facilities provide enough of each resource in the first category to meet the total demand. This is a necessary, but not a sufficient condition for feasibility. USMPM provides a lower bound and a set of facility locations for the primal subproblem.

IV. COMPUTATIONAL EXPERIENCE

A. DESCRIPTION OF THE PROBLEMS

Beasley [1990] provides a library of optimization problems that include capacitated facility location problems such as those used by Van Roy [1984] to test cross decomposition; however, many of these problems are infeasible for the CFLP with sole sourcing and the remaining problems are directly solved rapidly by CPLEX [1994]. We therefore generate more difficult random test problems of various sizes.

Our problem generation scheme is to specify a range of values for the data and to then randomly select the data value from a uniform distribution over this range. We first specify the number of customers and facilities (the problem name follows the convention, *e.g.*, 10C20 is ten facilities and twenty customers) and the minimum and maximum customer demand. The minimum supply for each facility is equal to the average demand times the average number of customers per facility. The maximum supply is equal to three times the minimum supply. Arbitrarily, the minimum fixed cost for each facility is equal to the supply, the maximum fixed cost is equal to twice the supply, the minimum variable cost for each customer and facility is equal to the customer demand, and the maximum variable cost is equal to twice the customer demand.

B. RESULTS AND ANALYSIS

Tables 1 to 14 below present results from applying the decomposition algorithms described in the previous chapter to our test problems. Each table contains a measure of the problem size (number of facilities and customers), the results for each algorithm and some general comments about the problem. The results for each algorithm include the time needed in processor (CPU) seconds on an IBM System 6000 Model 590H, the lower bound determined, the number of times that the various problems (SSDM in the DM column, SSDS in the DS column, PMLX and PMCX in the PM column, and PSLX and PSL in the PS column) are solved and remarks about the algorithm's performance. We attempt to run each algorithm until it converges to within 0.01 (the upper bound minus the lower bound all divided by the lower bound) of the best lower bound obtained by the algorithm. We truncate any algorithm that fails to converge after 7,200 CPU seconds or 250 iterations of any problem or 1,000 iterations of PSL.

We use CPLEX [1994] to obtain the optimal objective function value of the linear programming (LP) relaxation of the problem. The lower bounds obtained by the decomposition algorithm may be slightly less than the lower bound from the LP relaxation due to the convergence and truncation criteria for the decomposition algorithms described above. After a sufficient number of iterations, the decomposition algorithms will determine bounds that are equal to or better than the objective function value of the LP relaxation.

We also attempt to solve the problems with the CPLEX [1994] mixed integer solver to provide a basis for comparison. This commercial solver employs state of the art branch and bound techniques that include preprocessing, heuristic rounding to obtain the first integer solution, cut-off and shortcut techniques, clique and cover cuts, and numerous other features. We report the objective function value of the optimal solution or the best feasible solution for each problem. When the optimal solution is not known, we report the quality of the branch and bound solution in the remarks.

Problem: 10C20		Facilities: 10			Customers: 20		
Algorithm	**Time**	**Bound**	**Iterations**				**Remarks**
			DM	**DS**	**PM**	**PS**	
LP Relaxation		2,589.4					
BD6	0.52	2,628.6			6	7	
CD6	1.77	2,619.2	3	11	1	12	PMLX & PSLX converge
DW3	8.01	2,619.3	31	31			
GCD	272.91	2,643.0	5	13	6	14	(405 iterations in PSL) PMCX & PSL converge
GBD	148.12	2,660.0			8	9	(265 iterations in PSL)
CPLEX	3.84	2,662.0					optimal solution

Table 1. BD6 converges most rapidly and determines a good lower bound. CD6 performs well, but does not determine the best possible bound. GBD determines the best bound. GBD and GCD converge in a reasonable number of iterations, but their time performance is poor. PSL determines an upper bound in GBD and GCD that is equal to the optimal objective function value. All five decomposition algorithms find bounds that are better than the LP relaxation.

Problem: 10C35		Facilities: 10			Customers: 35		
Algorithm	**Time**	**Bound**	**Iterations**				**Remarks**
			DM	**DS**	**PM**	**PS**	
LP Relaxation		4,625.6					
BD6	1.00	4,699.0			8	9	
CD6	6.81	4,699.4	5	22	2	23	PMLX & PSLX converge
DW3	53.62	4,657.3	56	56			
GCD	2,396.6	4,661.5	5	14	7	15	(839 iterations in PSL) PMCX & PSL converge
GBD	2,589.21	4,662.8			15	15	(882 iterations in PSL)
CPLEX	8.56	4,705.0					optimal solution

Table 2. BD6 converges most rapidly and determines a good lower bound. CD6 performs well and determines the best lower bound. GBD and GCD converge in a reasonable number of iterations, but their time performance is poor. PSL determines an upper bound in GBD and GCD that is equal to the optimal objective value. All five decomposition algorithms find bounds that are better than the LP relaxation. CPLEX branch and bound performs well.

Problem: 15C30		Facilities: 15			Customers: 30		
Algorithm	**Time**	**Bound**	**Iterations**				**Remarks**
			DM	**DS**	**PM**	**PS**	
LP Relaxation		3,803.7					
BD6	64.59	3,845.4			68	69	
CD6	76.13	3,685.6	20	80	5	81	SSDS & PSLX converge
DW3	51.68	3,691.0	45	45			
GCD	5,263.52	3,614.0	16	38	17	39	(996 iterations in PSL) 0.034 convergence gap
GBD	4,613.95	3,641.0			31	32	(989 iterations in PSL) 0.076 convergence gap
CPLEX	607.10	3,916.0					optimal solution

Table 3. BD6 determines the best lower bound and is the only algorithm to improve the bound from the linear programming relaxation. CD6 does not obtain the same bound because PSLX and SSDS converge to within 0.01 first. DW3 converges to a bound in the least amount of time. PSL determines an upper bound in GBD and GCD that is equal to the optimal objective function value. GBD and GCD terminate before the problems converge to within 0.01 of the bound.

Problem: 15C50		Facilities: 15			Customers: 50		
Algorithm	**Time**	**Bound**	**Iterations**				**Remarks**
			DM	**DS**	**PM**	**PS**	
LP Relaxation		6,441.0					
BD6	9.22	6,428.0			19	20	
CD6	128.37	6,416.6	15	94	5	95	SSDM & PMLX converge
DW3	297.88	6,395.1	74	74			
CPLEX	6,091.50	6,960.0					solution within 0.072 of optimal

Table 4. BD6 converges rapidly to the best lower bound among the decomposition algorithms. None of the decomposition algorithms determine a bound that is better than the LP relaxation. This indicates that a convergence criteria of 0.01 truncates the algorithms too soon for this problem. GBD and GCD fail to converge or even provide reasonable bounds. CPLEX branch and bound obtains a feasible solution that is within 0.072 of optimal.

Problem: 20C40		Facilities: 20					Customers: 40
Algorithm	**Time**	**Bound**	**Iterations**				**Remarks**
			DM	**DS**	**PM**	**PS**	
LP Relaxation		4,618.8					
BD6	944.52	4,591.1			121	122	
CD6	244.65	4,546.9	20	148	7	149	SSDM & SSDS converge
DW3	172.41	4,548.0	57	57			
GCD	7,452.5	4,414.0	9	21	9	22	(635 iterations in PSL) 0.072 convergence gap
GBD	7,413.55	4,440.5			28	29	(620 iterations in PSL) 0.060 convergence gap
CPLEX	15,639.35	4,746.0					solution within 0.027 of optimal

Table 5. BD6 determines the best lower bound. CD6 and DW3 converge more rapidly to a weaker bound. No algorithm obtains a bound that is better than the LP relaxation of the problem. GBD and GCD terminate before they converge to within 0.01 of the bound.

Problem: 20C70		Facilities: 20					Customers: 70
Algorithm	**Time**	**Bound**	**Iterations**				**Remarks**
			DM	**DS**	**PM**	**PS**	
LP Relaxation		8,280.9					
BD6	370.57	8,323.6			87	88	
CD6	1,170.21	8,209.5	34	148	7	149	SSDS & SSDM converge
DW3	1,599.18	8,199.1	104	104			
CPLEX	9,784.0	8,730.0					solution within 0.046 of optimal

Table 6. BD6 converges rapidly to the best lower bound among the decomposition algorithms that converge. DW3 converges to the worse bound and requires more time that BD6 or CD6. CD6 converges to the same bound as DW3. If CD6 switches to BD6 when SSDS and SSDM converge instead of halting, it can detect the difference in the bounds and converge to the better of the two.

Problem: 25C30		Facilities: 25	Customers: 30				
Algorithm	**Time**	**Bound**	**Iterations**				**Remarks**
			DM	**DS**	**PM**	**PS**	
LP Relaxation		5,427.2					
BD6	23.97	5,404.9			23	24	
CD6	40.68	5,391.4	11	37	1	38	SSDM & SSDS converge
DW3	87.6	5,405.0	38	38			
GCD	3290.85	5,410.3	16	33	12	34	(226 iterations in PSL) SSDM & SSDS converge
GBD	7284.57	5,406.0			100	101	(454 iterations in PSL) 0.063 convergence gap
CPLEX	4,865.53	6,366.0					solution within 0.146 of optimal

Table 7. GCD determines the best lower bound. BD6, CD6 and DW3 converge more rapidly to a similar bound. No algorithm obtains a bound that is better than the LP relaxation of the problem. GBD halts before they converge to within 0.01 of bound. Neither GBD nor GCD determines a good lower bound.

Problem: 25C45		Facilities: 25	Customers: 45				
Algorithm	**Time**	**Bound**	**Iterations**				**Remarks**
			DM	**DS**	**PM**	**PS**	
LP Relaxation		9,141.5					
BD6	0.45	9,217.3			1	2	
CD6	201.72	9,297.7	23	24	0	2	SSDM & SSDS converge
DW3	94.14	9,315.1	52	52			
GCD	4,553.20	9,483.5	5	10	3	11	(212 iterations in PSL) PMCX & SSDM converge
GBD	7,194.01	9,572.4			24	25	(428 iterations in PSL)
CPLEX	8,452.92	10,503.0					solution within 0.128 of optimal

Table 8. GBD determines the best lower bound. GCD converges to a similar bound in much less time. Every algorithm obtains a bound that is better than the LP relaxation of the problem. CD6 determines after two iterations which decomposition algorithm converges to the better bound and switches to DW3.

Problem: 25C60		Facilities: 25			Customers: 60		
Algorithm	**Time**	**Bound**	**Iterations**				**Remarks**
			DM	**DS**	**PM**	**PS**	
LP Relaxation		10,894.9					
BD6	1.32	10,835.0			2	3	
CD6	2.33	10,805.7	0	2	0	3	SSDS & PSLX converge
DW3	743.53	10,850.5	77	7			
GCD	7,418.13	10,901.0	7	14	7	15	(287 iterations in PSL) 0.019 convergence gap
GBD	7,568.60	10,927.0			15	16	(287 iterations in PSL) 0.017 convergence gap
CPLEX	3,916.95	12,312.0					solution within 0.126 of optimal

Table 9. GBD and GCD determine the best lower bounds, but do not converge. DW3 converges to a better bound in less time. BD6 and CD6 converge very rapidly to similar bounds.

Problem: 30C35		Facilities: 30			Customers: 35		
Algorithm	**Time**	**Bound**	**Iterations**				**Remarks**
			DM	**DS**	**PM**	**PS**	
LP Relaxation		7,059.1					
BD6	0.73	7,053.3			1	2	
CD6	1.49	7,082.7	0	1	0	2	SSDS & PSLX converge
DW3	656.60	7,135.7	41	41			
GCD	3,556.45	7,125.0	12	24	0	25	(236 iterations in PSL) SSDS & SSDM converge
GBD	7,360.97	7,197.0			85	86	(467 iterations in PSL) 0.10 convergence gap
CPLEX	7,396.27	8,253.0					solution within 0.115 of optimal

Table 10. GBD determines the best lower bound, but does not converge. GCD converges to a different bound in much less time. DW3 converges to the same bound as GCD in even less time. BD6 and CD6 are extremely fast, but converge to weaker bounds.

Problem: 30C50		Facilities: 30			Customers: 50		
Algorithm	Time	Bound	Iterations				Remarks
			DM	DS	PM	PS	
LP Relaxation		10,458.1					
BD6	0.68	10,481.7			1	2	
CD6	219.98	10,669.4	24	25	0	2	SSDS & SSDM converge
DW3	664.26	10,700.4	57	57			
GCD	6,011.74	10,826.8	6	12	3	13	(285 iterations in PSL) PMCX & SSDM converge
GBD	7,473.08	10,839.8			13	14	(362 iterations in PSL) 0.037 convergence gap
CPLEX	7,200.21	11,527.0					solution within 0.082 of optimal

Table 11. GBD determines the best lower bound, but does not converge. GCD converges to the same bound in slightly less time. DW3 converges to a good bound in much less time, but CD6 obtains the same bound even faster. CD6 again detects the best algorithm and switches to DW3 after two iterations. BD6 is extremely fast, but converges to the weakest bound.

Problem: 30C75		Facilities: 30			Customers: 75		
Algorithm	Time	Bound	Iterations				Remarks
			DM	DS	PM	PS	
LP Relaxation		13,651.0					
BD6	3.93	13,539.7			3	4	
CD6	43.89	13,583.5	4	16	0	17	SSDS & PSLX converge
DW3	1,478.32	13,586.6	92	92			
CPLEX	7,200.24	15,379.0					solution within 0.123 of optimal

Table 12. DW3, CD6 and BD6 all converge to similar bounds. BD6 converges rapidly. CD6 performs well. DW3 does not perform as well. GBD and GCD fail to obtain usable bounds.

Problem: 40C50		Facilities: 40			Customers: 50		
Algorithm	**Time**	**Bound**	**Iterations**				**Remarks**
			DM	**DS**	**PM**	**PS**	
LP Relaxation		9,386.7					
BD6	6.74	9,314.9			4	5	
CD6	136.00	9,342.1	6	41	1	42	PMLX & PSLX converge
DW3	678.99	9,375.6	60	60			
GCD	8,598.28	9,324.0	3	7	2	8	(230 iterations in PSL) 0.074 convergence gap
GBD	7,434.07	9,332.0			7	8	(227 iterations in PSL) 0.073 convergence gap
CPLEX	7,200.20	10,867.0					solution within 0.147 of optimal

Table 13. DW3 determines the best lower bound. BD6 and CD6 converge more rapidly converged to a similar bound. GCD and GBD do not converge.

Problem: 40C75		Facilities: 40			Customers: 75		
Algorithm	**Time**	**Bound**	**Iterations**				**Remarks**
			DM	**DS**	**PM**	**PS**	
LP Relaxation		14,335.6					
BD6	1.84	14,321.2			1	2	
CD6	24.39	14,354.7	0	1	0	2	SSDS & PSLX converge
DW3	4016.75	14,396.8	83	83			
GCD	9,434.05	14,446.6	1	2	0	3	(145 iterations in PSL) 0.063 convergence gap
GBD	9,649.32	14,447.6			2	3	(148 iterations in PSL) 0.082 convergence gap
CPLEX	7,200.34	16,119.0					solution within 0.114 of optimal

Table 14. GBD and GCD determine the best lower bound, but do not converge. DW3 converges to a good bound, but is also very slow. CD6 and BD6 both obtain reasonable bounds very rapidly.

BD6 converges to a lower bound rapidly for all problems. DW3 is too slow to converge to a lower bound for some problems. DW3 performs poorly for larger problems and problems with smaller convergence tolerances because later iterations between SSDS and SSDM required significantly more time to solve than previous iterations.

CD6 converges to a lower bound quickly for most problems. It converges rapidly enough that it is possible to alter the halting criteria to obtain a better bound. For example, CD6 can continue for some number of iterations beyond the convergence of PSLX and PMLX and possibly determine whether DW3 will converge to a better bound.

GBD and GCD usually converge to a lower bound too slowly, but they often do produce good bounds in a reasonable number of iterations. GBD and GCD solve PSL by Danzig-Wolfe decomposition at every iteration and this requires too much time.

C. USING THE LOWER BOUND

Tables 15 to 18 below present results from using the decomposition problems to produce a feasible solution to the CFLPSS for some of the previous problems. To obtain this solution we use the CPLEX branch and bound algorithm to solve the problem obtained by fixing the facility decisions to the configuration last determined by the decomposition algorithm and setting the lower cut-off to the lower bound determined by the decomposition algorithm. For each algorithm the table contains the added time in seconds to obtain the solution, the total time in seconds, the objective function value, and the quality of the solution measured against the lower bound determined by that decomposition algorithm.

Problem: 10C20			Facilities: 10		Customers: 20
	Time		**Solution**		
Algorithm	**Added**	**Total**	**Value**	**Quality**	**Remarks**
BD6	0.37	0.89	2,791	0.062	
CD6	0.46	2.23	2,841	0.085	
DW3	0.28	8.59	2,789	0.065	
GCD	0.10	273.01	2,724	0.031	
GBD	0.10	148.22	2,724	0.024	
CPLEX		3.84	2,662.0	0.000	

Table 15. All algorithms determine feasible solutions with objective function values that are within at least 0.085 of the optimal objective function value in significantly less time than branch and bound.

Problem: 10C35			Facilities: 10		Customers: 35
	Time		**Solution**		
Algorithm	**Added**	**Total**	**Value**	**Quality**	**Remarks**
BD6	337.57	338.57	4,822	0.026	2nd problem
CD6	0.81	7.62	4,840	0.030	
DW3	0.78	54.40	4,787	0.028	
GCD	0.77	2,397.37	4,903	0.051	
GBD	1.55	2,590.76	5,015	0.076	
CPLEX		8.56	4,705	0.000	

Table 16. All algorithms determine feasible solutions. The first problem solved by BD6 is not feasible. CD6 determines a solutions with an objective function value that are within at least 0.03 of the optimal objective function value in less time than branch and bound.

Problem: 15C30			Facilities: 15		Customers: 30
	Time		**Solution**		**Remarks**
Algorithm	**Added**	**Total**	**Value**	**Quality**	
BD6	8.13	72.72	3,969	0.030	
CD6	5.46	81.59	4,106	0.114	
DW3	85.34	137.02	4,051	0.097	
CPLEX		607.61	3,916	0.000	

Table 17. GBD and GCD did not determine feasible solutions. All other algorithms did determine feasible solutions faster than branch and bound, but the quality of these solutions is not always good.

Problem: 15C50			Facilities: 15		Customers: 50
	Time		**Solution**		**Remarks**
Algorithm	**Added**	**Total**	**Value**	**Quality**	
BD6	3,783.29	3,792.51	6,606	0.028	6th problem
CD6	33.75	162.12	6,590	0.027	
DW3	2,867.98	3,165.86	6,651	0.040	4th problem
CPLEX		6,091.50	6,960	0.072	

Table 18. All three algorithms determine feasible solutions that were better than the best known branch and bound solution. CD6 performed significantly better than all other algorithms finding a solution with an optimal objective function value within 0.027 of optimal in less than three minutes.

This procedure to find a feasible solution for the CFLPSS does not always work well for the larger problems in this study. Computational testing indicates that it is often necessary to solve the problem several times with different facility configurations to find a feasible solution and in many cases no feasible solution is found. A better approach may be to check for feasibility each time customers are assigned to facilities during the decomposition algorithm and retain feasible facility configurations.

D. A GRAPHICAL EXAMPLE OF CONVERGENCE

In this section we graphically compare the convergence of cross decomposition (CD6) with Benders decomposition (BD6) and Danzig-Wolfe decomposition (DW3) for problem 20C40 shown in Table 5. For the problems tested the convergence characteristics of these decomposition methods on this problem are typical. In figures two through four below, the y-axis is labeled with objective function value and the x-axis is labeled with the CPU time at the completion of each iteration of the decomposition algorithm. We plot the best bound determined by the decomposition problem at each iteration.

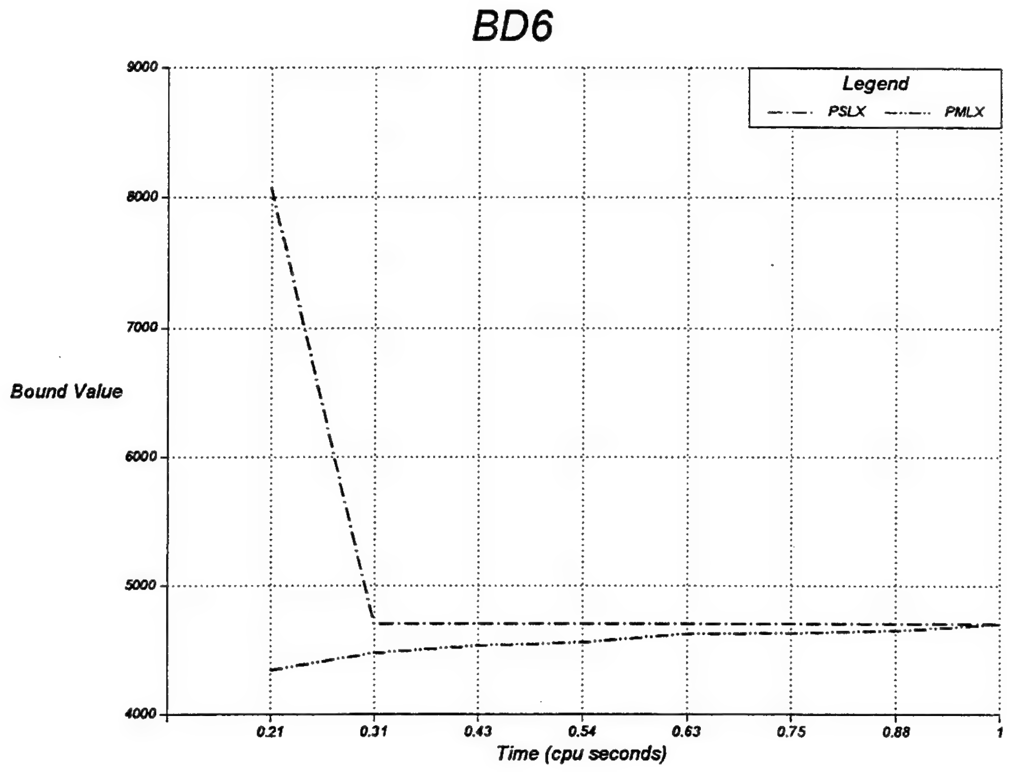

Figure 2. We solve PMLX and PSLX at every iteration. PSLX rapidly determines a tight upper bound and does not improve this bound after the second iteration. PMLX gathers cuts and improves the lower bound steadily at every iteration. BD6 halts when the objective function value of PMLX is within 0.01 of the upper bound determined by PSLX.

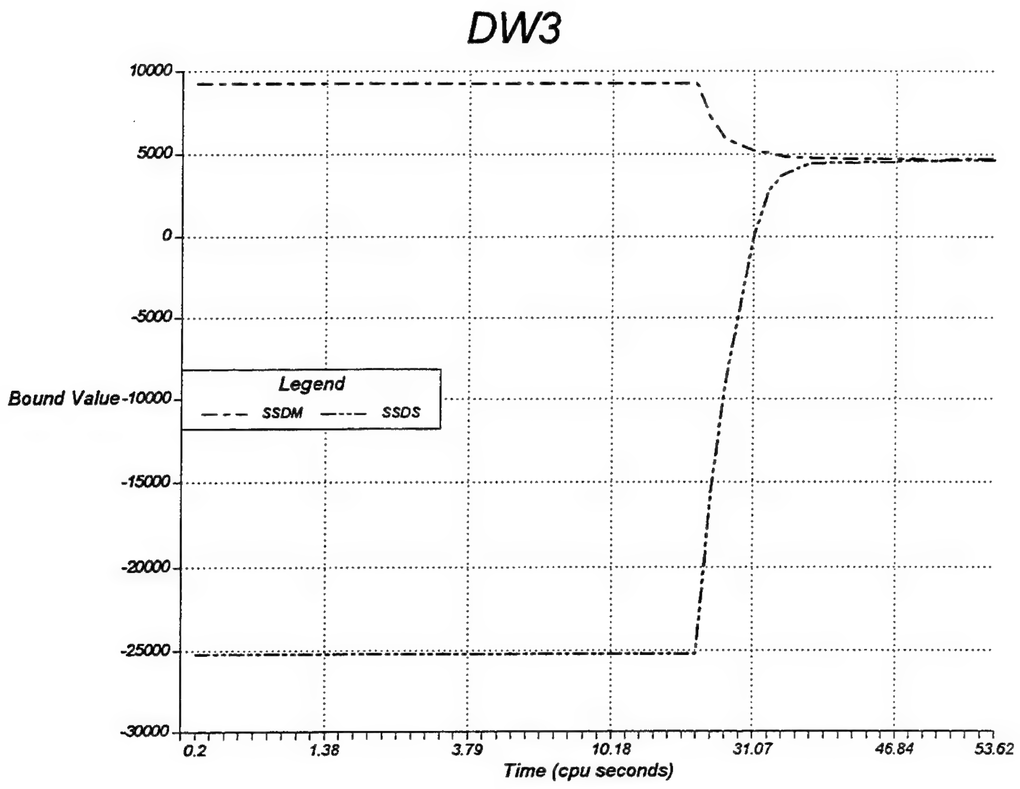

Figure 3. DW3 solves SSDM and SSDS at every iteration. These problems do not provide good bounds initially, but after thirty-five iterations without progress they begin to converge. Note that the time to solve each iteration increases markedly as the decomposition algorithm progresses. The first thirty iterations require less than half as much time as the next ten iterations. The last sixteen iterations account for over forty percent of the total solution time. SSDS determines a good upper bound after forty iterations and only improves slightly after that point. SSDM gathers cuts and improves steadily after forty iterations. DW3 halts when the objective function value of SSDM is within 0.01 of the best upper bound determined by SSDS.

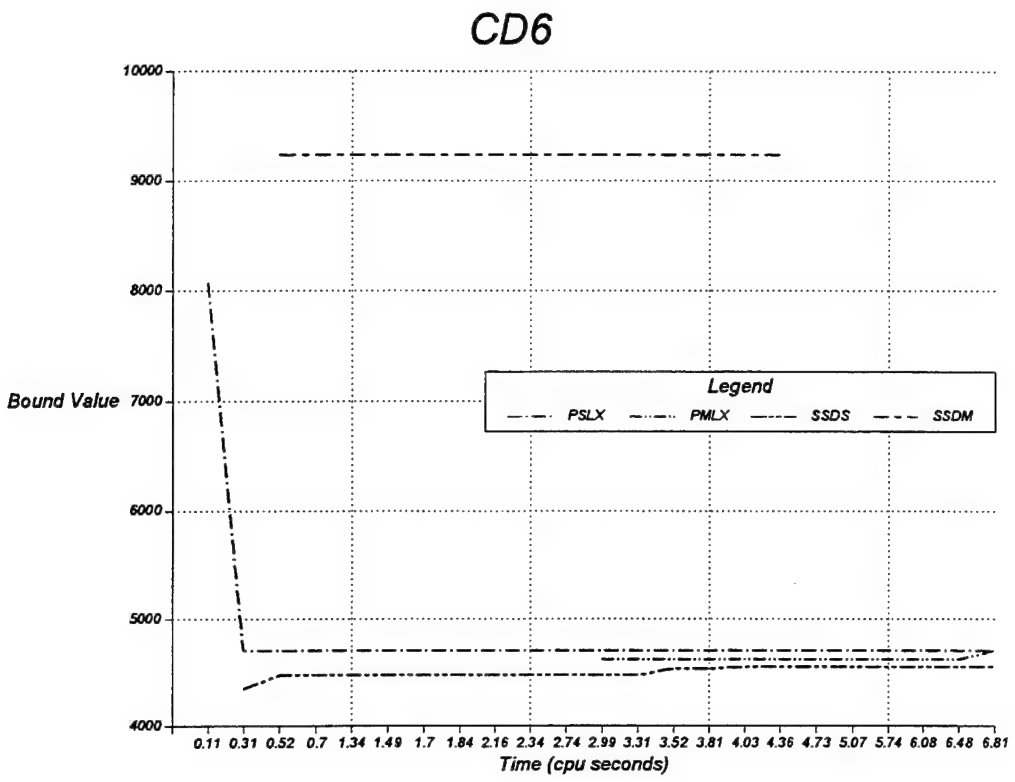

Figure 4. CD6 solves SSDS and PSLX at every iteration. These decomposition problems determine moderately good bounds initially, but do not converge rapidly. PSLX never improves the upper bound after the second iteration. When a convergence test fails, a master problem is solved. SSDM is solved more frequently than PMLX, but this problem never provides a good upper bound. PMLX provides a good lower bound when first solved on the 12th iteration. CD6 halts when the objective function value of PMLX converges with the upper bound determined by PSLX.

It is possible that the lower bound determined by DW3 is superior to the bound determined by CD6 for this problem. To determine if this is the case we would iterate between SSDM and SSDS after PSLX and PMLX converged until either they converged or the bound from PMLX exceeded the bound from SSDM.

E. COMPUATATIONAL EXPERIENCE WITH USM

1. Description of Problems

We generate random test problems of various sizes for the unit stationing problem. The problem generation scheme is similar to that described previously for CFLPSS with slight modifications for multiple resources. We first specify the number of units and facilities (the naming convention is *e.g.*, 24U36 is 24 facilities and 36 units) and the minimum and maximum unit demand. The minimum supply of resources of type R_0 (those resources not shared by units) for each facility is equal to the average demand times the average number of units per facility. The maximum supply of resources of type R_0 is equal to three times the minimum supply. The minimum supply of resources of type R_1 (those resources shared by units) for each facility is equal to the minimum demand. The maximum supply of resources of type R_1 is equal to the maximum demand. Arbitrarily, the minimum fixed cost for each facility is equal to the maximum supply among the resource categories, the maximum fixed cost is equal to twice the minimum fixed cost, the minimum variable cost for each unit and facility is equal to the unit demand, and the maximum variable cost is equal to twice the unit demand.

2. Results

Table 19 below presents results from the Benders decomposition algorithm (BD6) applied to the unit stationing model, the branch and bound algorithm applied to the problem obtained by fixing the facility configuration to that of the last iteration of BD6, the branch and bound algorithm applied to the unit stationing model (USM), and the simplex algorithm applied to the linear programming relaxation of the unit stationing model. For each problem the table contains for BD6 the total time in seconds, the lower bound, and the number of iterations for BD6 to converge; for the problem to assign units to the facilities selected by BD6 the total time in seconds and the objective function value; for branch and bound the total time in seconds and the objective function value; and for the linear programming relaxation the objective function value.

Problem	BD6			Assign		CPLEX		LP Relaxation
	Time	Bound	Iterations	Time	Solution	Time	Solution	
18U24	19.30	4,873	53	520.0	5,826	1,800	5,585	4,682.5
18U36	442.10	8,215	232	580.7	9,330	1,800	9,818	7,737.9
24U36	48.11	6,574	41	–	–	1,800	7,571	6,543.9
24U48	99.21	9,651.8	43	1303.8	10,809	1,800	11,774	9,419.5
– indicates that no feasible integer solution is found.								

Table 19. BD6 converges rapidly and determines a good lower bound for all problems except 18U36 . BD6 finds bounds for all problems that are better than the LP relaxation. BD6 finds feasible integer solutions for all problems except 24U36. All CPLEX branch and bound solutions are the best known integer solution.

These results indicate that a Benders decomposition may be a good alternative to branch and bound for finding good feasible solutions to USM.

V. CONCLUSIONS

A. PERFORMANCE OF THE DECOMPOSITION ALGORITHMS

Benders decomposition (BD6) is the best algorithm, especially for larger problems, because it converges to a lower bound most rapidly. Lagrangean relaxation (DW3) performs well for some small problems, but later iterations of this algorithm require significantly more time to solve than the earlier iterations. Cross decomposition (CD6) is a good algorithm for medium sized problems. It converges rapidly enough that it may be possible to continue the algorithm until it is clear whether BD6 or DW3 will converge to the better bound.

Generalized Benders decomposition (GBD) and generalized cross decomposition (GCD) are too slow, but they should not be abandoned completely. They are guaranteed to converge to the best possible lower bound and they do converge in a reasonable number of iterations. The primary problem with GCD and GBD is the poor performance of PSL. The next section discusses techniques to address this shortcoming.

B. TOPICS FOR FURTHER RESEARCH

1. Determining Upper Bounds

The decomposition algorithms presented in this thesis find lower bounds on the optimal objective function value. In the process they may also identify a feasible set of facilities. The lower bound is useful to judge the quality of a feasible solution. Fixing the facility variables determined in the last iteration of the decomposition algorithm and solving the resulting problem is typically an efficient method to obtain a feasible solution. This method fails if the set of facilities does not contain a feasible set of customer assignments. Other methods for obtain upper bounds and feasible solutions should be explored.

2. Improving the Efficiency of the Algorithms

It is possible to improve the efficiency of the current implementations in several ways. For the decomposition algorithms considered in this thesis there are techniques to strengthen the cuts currently determined in the subproblems. Application of these techniques would reduce the number of iterations required to achieve convergence and may prove computationaly attractive.

As implemented for this thesis, GBD and GCD are not viable solution methods. However, they are theoretically very attractive because they converge to the best bound that can be obtained from the decomposition algorithms we considered. In GBD and GCD the most expensive problem is PSL. Three alternate solution approaches for PSL should be considered.

(1) Use a heuristic method (such as solving a relaxed problem and rounding to an integer solution) to solve $PSDS_i$ until $PSDS_i$ and DML near convergence, then switch to an optimal solution method.

(2) Solve the binary knapsack problems in PSL with a pseudo-polynomial algorithm (such as dynamic programming). These methods are often superior to branch and bound.

(3) Solve PSL using subgradient optimization instead of Danzig-Wolfe decomposition.

3. Implementing the Stationing Model

The motivation for this thesis research was the unit stationing model. The Benders decomposition method for the stationing model outlined in this thesis can be extended to cross decomposition by implementing a Lagrangean relaxation algorithm. Computational experiments should be performed with these decomposition algorithms on a larger set of problems.

4. Improving the Experimental Design

The computational results reported in this thesis are for a limited number of instances of each problem size and for one set of parameters that generate the problems. It would be better to test the algorithms against a random sample of problem instances for each problem size. It would also be useful to generate problems with different sets of parameters. This may lead to additional work to establish a set of metrics for CFLP problems and to quantify the difficulty of problems in terms of these metrics.

APPENDIX. BENDERS DECOMPOSITION

In this section we derive the Benders master problem for the capacitated facility location problem (CFLP). In Benders decomposition we identify complicating variables that can be fixed to produce a simpler problem. For the CFLP the binary variables (y) associated with the facility location decisions complicate the problem. With the y variables fixed the CFLP is a simple linear program: a bipartite network with gains that provides a minimum cost assignment of customer demand to facilities.

The problem below is a reformulation of the CFLP with an outer optimization over the complicating variables (y), and an inner optimization over the simple variables (x).

$$\underset{y}{\text{minimize}} \quad \sum_i f_i \cdot y_i + \left\{ \begin{array}{lll} \underset{x}{\text{minimize}} & \sum_{i,j} c_{i,j} \cdot x_{i,j} & \\ & & \\ \text{subject to:} & \sum_i x_{i,j} = 1 \quad [\lambda_j] & \forall j \\ & \sum_j d_j \cdot x_{i,j} \le s_i \cdot y_i \quad [\mu_i] & \forall i \\ & x_{i,j} \le y_i \quad [\nu_{i,j}] & \forall i,j \\ & x_{i,j} \ge 0 \quad \forall i,j & \end{array} \right\}$$

$$\text{subject to:} \quad \sum_i s_i \cdot y_i \ge \sum_j d_j$$

$$y_i \in \{0,1\} \quad \forall i$$

The added constraint in the outer optimization ensures that the facility configuration is feasible.

With the facility configuration fixed we replace the inner optimization with its linear programming dual:

$$\underset{y}{\text{minimize}} \quad \sum_i f_i \cdot y_i + \left\{ \begin{array}{ll} \underset{\lambda,\mu,\nu}{\text{maximize}} & \sum_j \lambda_j - \sum_i s_i \cdot y_i \cdot \mu_i - \sum_{i,j} y_i \cdot \nu_{i,j} \\ & \\ \text{subject to:} & \lambda_j - d_j \cdot \mu_i - \nu_{i,j} \le c_{i,j} \quad [x_{i,j}] \quad \forall i,j \\ & \mu_i \ge 0 \quad \forall i \qquad \nu_{i,j} \ge 0 \quad \forall i,j \end{array} \right\}$$

$$\text{subject to:} \quad \sum_i s_i \cdot y_i \ge \sum_j d_j$$

$$y_i \in \{0,1\} \quad \forall i$$

Let

$$T = \left\{ t \mid \left(\lambda^{(t)}, \mu^{(t)}, \nu^{(t)}\right) \text{ is a (dual) feasible extreme point solution} \right\}$$

be the index set of all (dual) feasible extreme point solutions of the inner optimization problem above. Then the previous problem can be written as

$$\underset{y}{\text{minimize}} \quad \left\{ \underset{t \in T}{\text{maximize}} \quad \sum_j \lambda_j^{(t)} + \sum_i \left(f_i - s_i \cdot \mu_i^{(t)} - \sum_j \nu_{i,j}^{(t)} \right) \cdot y_i \right\}$$

$$\text{subject to:} \quad \sum_i s_i \cdot y_i \ge \sum_j d_j$$

$$y_i \in \{0,1\} \quad \forall i$$

This problem is equivalent to the original problem since the dual of the inner optimization problem attains its optimal solution at one of a finite number of extreme points. (The dual problem is bounded if the primal problem is feasible.) The problem above can also be written as the mixed integer program below.

$$\underset{\rho, y}{\text{minimize}} \quad \rho$$

$$\text{subject to:} \quad \rho \geq \sum_j \lambda_j^{(t)} + \sum_i \left(f_i - s_i \cdot \mu_i^{(t)} - \sum_j \nu_{i,j}^{(t)} \right) \cdot y_i \qquad \forall t \in T$$

$$\sum_i s_i \cdot y_i \geq \sum_j d_j$$

$$y_i \in \{0,1\} \quad \forall i$$

The Benders master problem is a relaxation of this problem obtained when only a subset of the constraints associated with the index set *T* are known. The Benders master problem provides a feasible facility configuration and a lower bound on the optimal objective function value of the original problem.

The set *T* is usually large and only a subset of these constraints are binding in the optimal solution. The strategy adopted by Benders decomposition is to solve a relaxed master problem that contains a subset of these constraints to obtain a feasible facility configuration and to then solve another problem, the Benders subproblem, to determine if a lower cost facility configuration exists. If the facility configuration from the relaxed Benders master problem is not optimal, the subproblem provides a new constraint that is violated by this facility configuration. This new constraint, called a Benders cut, is one of the constraints in the index set *T* that is not already in the master problem. The problem below is the Benders subproblem.

$$\underset{x}{\text{minimize}} \quad \sum_{i} f_i \cdot \hat{y}_i + \sum_{i,j} c_{i,j} \cdot x_{i,j}$$

subject to:

$$\sum_{i} x_{i,j} = 1 \quad [\lambda_j] \quad \forall j$$

$$\sum_{j} d_j \cdot x_{i,j} \le s_i \cdot \hat{y}_i \quad [\mu_i] \quad \forall i$$

$$x_{i,j} \le \hat{y}_i \quad [v_{i,j}] \quad \forall i,j$$

$$x_{i,j} \ge 0 \quad \forall i,j$$

This problem is identical to the inner optimization problem in the reformulation of the CFLP. The Benders subproblem is a restriction of the CFLP obtained by fixing the facility locations. The optimal solution to the Benders subproblem provides a set of dual variable values that form a Benders cut and an upper bound on the optimal objective function value of the original problem. The objective function value of the Benders subproblem is equal to the optimal solution of the original problem when the facility configuration is optimal.

Benders decomposition starts with a feasible facility configuration and iterates between the Benders subproblem and the Benders master problem. At each iteration, if the facility configuration is not optimal, the subproblem provides a new Benders cut. This cut is a violated constraint from the constraints associated with the index set *T*. In the extreme case, after a finite number of iterations the subproblem produces all of the constraints associated with the index set *T* and the master problem is equivalent to the original problem. Thus, after a finite number of iterations the master problem and the subproblem must converge in objective function value to the optimal objective function value of the original problem (CFLP).

LIST OF REFERENCES

[Beasley 1990]	BEASLEY, J. E. OR-Library: Distributing Test Problems by Electronic Mail. *Journal of the Operational Research Society*. 41 (11), 1069-1072.
[BRAC 1990]	The Defense Base Closure and Realignment Act of 1990. Title XXIX of United States Public Law 101-510.
[Brooke *et. al.* 1992]	BROOKE, A., D. KENDRICK and A. MEERAUS. GAMS, A User's Guide. The Scientific Press. South San Francisco, CA.
[Cornuejols *et. al.* 1991]	CORNUEJOLS, G., R. SRIDHARAN and J.M. THIZY, 1991. A Comparison of Heuristics and Relaxations for the Capacitated Plant Location Problem. *European Journal of Operations Research*. 50, 280-297.
[CPLEX 1994]	Using the CPLEX Callable Library. CPLEX Optimization, Inc. Copyright 1989-1994. Incline Village, NV.
[Dell *et. al.* 1994]	DELL, R.F., C. FLETCHER, S.H. PARRY, R.E. ROSENTHAL. Modeling Army Maneuver and Training Base Realignment and Closure. Technical Report NPS-OR-94-002, Naval Postgraduate School, Monterey, CA.
[Holmberg 1990]	HOLMBERG, K. On the Convergence of Cross Decomposition. *Mathematical Programming*. 47, 269-296.
[Holmberg 1994]	HOLMBERG, K. Cross Decomposition Applied to Integer Programming Problems: Duality Gaps and Convexification in Parts. *Operations Research*. 42, 657-668.
[Magnanti & Wong 1990]	MAGNANTI, T.L. and R.T. WONG. Decomposition Methods for Facility Location Problems. From Discrete Location Theory edited by P.B. Mirchandani and R.L. Francis. John Wiley and Sons, Inc. New York, NY.

[Nemhauser & Wolsey 1988]	NEMHAUSER, G.L. and L.A. WOLSEY. Integer and Combinatorial Optimization. John Wiley and Sons. New York, NY.
[Van Roy 1983]	VAN ROY, T.J. Cross Decomposition for Mixed Integer Programming. *Mathematical Programming.* 25, 46-63.
[Van Roy 1986]	VAN ROY, T.J. A Cross Decomposition Algorithm for Capacitated Facility Location. *Operations Research.* 34, 145-163.
[Sunset Technology 1987]	XA: Professional Linear Programming System. Sunset Software Technology. San Marino, CA.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Fort Belvoir, Virginia 22060-6218	2
2.	Library, Code 13 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Professor Robert Dell Code OR/De Naval Postgraduate School Monterey, California 93943-5002	4
3.	Professor Gerald Brown Code OR/Bw Naval Postgraduate School Monterey, California 93943-5002	1
4.	Professor Dave Morton Department of Mechanical Engineering Engineering Teaching Center The University of Texas at Austin Austin, Texas 78712	1
5.	Director, TRAC ATTN: ATRC - FA Ft. Leavenworth, Kansas 66027	1
6.	Director, TRAC ATTN: MAJ Leroy A. Jackson P.O. Box 8692 Monterey, California 93943-0692	3